Project FlowKap Documentation

By Team Wolfpack
Alison Tai E'15 & Hyung-seo Park E'15
Advisors: Professor Remco Chang & Professor Chris Gregg

Table of Contents

Getting Started

This getting started guide assumes that you are hoping to reboot the project, and are hoping to get it up and running again. If you are just looking to see it in action, you may find it hosted with the Tufts University data from the Fall 2014 semester at flowkap.azurewebsites.net.

Technical Pre-Requisites: You will need to have Git installed, and internet access.

Background: Our senior capstone project's goal was to visualize the flow of students from one building to another in order to correlate it with departmental growth.

Viewing the Project:

1. Open up a terminal and navigate to the folder you would like to keep the project. Run: **git clone https://github.com/alictai/wolfpak.git**. You should now have all of the code required for the project.
2. Open up **index.html** in a browser and you should see the project as Team Wolfpack left it.

Data Format:

1. In the git clone, you should see a Data folder. If you have received data from the administration that looks like the data in **Room Capacities_Computer Science Project.xls**, then these directions are for you.
2. Save all data that you have as a CSV. This must be done in order for the Python scripts to work on it.
3. Cleaning up the building data. Only follow these steps if the rooms on campus have changed since this project was last worked on.
    a. In the Data Folder you will find a file called **clean_data.py.** Open this file and scroll down to the main() function.
    b. Replace where it says 'Building_Code_List_Computer_Science Project' with the file you were given that indicates the building codes on campus and the date/time it was generated at (which should be found in the file you were given).
    c. Replace 'Room_Capacities_Computer_Science_Project.csv' with the file you were given that indicates the capacities of all of the rooms on campus.
    d. Make sure all of your data is in the same folder as the script. Run the script from the command line by running **python clean_data.py**.

   e. This should create a file called **Rooms.csv** which has the data neatly parsed into the columns Building, Code, Room, Room_Code, Description, and Capacity.

4. Cleaning up the course data.
   a. In the Data folder, you will find a file called **clean_course_data.py**. Open this file and scroll down to the main() function.
   b. Replace 'Fall 2014 Course List and Locations_Computer Science Project.txt' with the name of your course list file.
   c. Run **python clean_course_data.py**. This should create a file called **CourseLocations.csv** which has the data neatly parsed as needed for step 5.
5. Combining the course and location data.
   a. In the data folder, you will find a file called **roomlookup.py**.
   b. If you have changed any of the output file names, then you must make sure they match in the main() function of this file.
   c. Run **python roomlookup.py**. This should create a file called **CourseLocations2.csv** which has each course appended with information about the building it is located in.
6. Finally, **CourseLocations2.csv** contains the data which the application will understand.

Note: In the case that the data formats of the aforementioned information are in a diferent format, these are the columns which are required for the application (CourseLocations2.csv):

- id - arbitrarily assigned, unique
- Course - the course code
- Title - the course name
- CourseType - how the course is taught (LEC - lecture, LAB - laboratory, SEM - seminar, etc)
- Instructor - the name of the course instructor with no commas, in the format [last name] [optional middle initial][first name]
- StartDate - the day the course began
- EndDate - the day the course ended
- Weekdays - what days of the week the class took place
- StartTime - what time the course classes began
- EndTime - what time the course classes ended
- Location - the building code and room number (separated by a space) of the class
- RoomType - how the room is setup
- Enrolled - the number of students enrolled in the class
- Credits - how many credits the class is worth

- Department - the department which the class is under
- BuildingName - the name of the building which the class took place in
- RoomName - the name of the building appended with a space and then the room number, which the class took place in

Database Setup:

1. In the original version of FlowKap, we ran an Azure SQL database that served up the data as necessary. Unfortunately, we could not financially continue to host the database so we took out the connection, but the PHP which we originally used for this can be found in the git folder.
2. During testing, we worked with a string-dumped version of all of the data which can be seen in **scripts/CourseLocationCSV.js**. The string found saved as **courseLocations** can be replaced with the contents of **data/CourseLocations2.csv** in order to get the prototype up and running.

We strongly suggest not going our way (SQL), and finding a more scalable approach to hosting the data (NoSQL) as appropriate.

Administrative Contacts & Information:

Chris Gregg - Senior Capstone Professor
Remco Chang - Senior Capstone Advisor

Your Senior Capstone professor and advisor are your main contacts within the Computer Science Department who know thoroughly what was going on with your project. They are your best contacts who can vouch for you when it came to asking for data. Remember to CC them on all emails with administration.

Dean Paul Stanton - Tufts University Dean of Student Services
Nicole Repucci - Tufts University Academic & Event Scheduling Coordinator

Dean Stanton was our ticket into all of the data we received. He met with us about our project, and happily asked different members of the administration for the data that we needed. In turn, on request from Dean Stanton, Nicole Repucci met with us to provide us with the Tufts course data which is kept in EMS - and is the source of all of our data.

Lois Stanley - Tufts University Director of Campus Planning
Tyler Patrick - Consultant from Sasaki
Brian Irwin - Consultant from Sasaki

Tufts University recently worked with a classroom distribution consulting firm, Sasaki, which uses technology for campus planning and Lois Stanley was their Tufts University contact. They sat down with Hyung and I, and loved the project - so I’m sure they would like to continue receiving updates.

Technical Details

- For the visualization we used **D3**. D3 is essentially a javascript library that offers some powerful tools for creating beautiful graphics. However, it is a little tricky to use. Much of it involves tacking on additional D3 elements (path, chords etc.) on the html and modifying its style and attributes. A lot of what we did (and presumably what you would do) involved taking existing D3 visualization code (available readily online... there is literally almost everything you could imagine) and tailoring them to our needs and wants. To extend our project I would just advise that you first learn the basics of D3, understand (just generally) how our project's D3 code works and then move on to add your own visualizations.
- To host our project, we used **Microsoft Azure**. This also allowed us to use an online database to store our data. We used Microsoft Azure because we are both going to be MS employees next year and we had access to a free Azure server that could host our project. Though before considering hosting the project, I would advise to test things locally first. You don't have to use Azure to host the website, anything else would do. Ideally choose an option that gives you a database.
- **Developing locally** can be tricky if you are trying to read from a CSV since you can't access local files from a browser. Neither can you just host on a github page because it then runs into some cross domain referencing issues.

Project Files:

- **index.html:** this is the only web page that we have. Open this in your browser to see our project.
- **bundle.js:** D3 code that controls the bundle visualization (the segmented circle with arcs in-between)
- **heatmap.js:** D3 code that controls the heatmap that pops up when use clicks on a circle segment
- **piechat.js:** D3 code that controls the piechart that pops up next to the heatmap
- **parser.js:** js code used to parse and process data to be readied for the visualization
- **ui.js:** js code that controls some UI interactions (buttons, resize etc.)

Potential Improvements

1. Establish a better way of acquiring student data. After providing a prototype, it is much easier to crowdsource data and it’s then possible to create a more accurate visualization of student flow.
2. Provide animation functionality to allow a user to follow student flow throughout the day instead of viewing a full summary at once.
3. Create more filters based on year or any additional data. If crowdsourced data is collected, it would be very useful to map out flow based on majors instead of department.
4. Add tools for administration to support optimal classroom allocation - moving from an exploratory visualization to a tool.
    a. Functionality to establish what-if scenarios.
    b. Functionality to display individual mileage based on class choice.
5. Restructure database and figure out what results should show in the event that the data is scaled to more schools or semesters.
6. Make the web application mobile friendly. It is usable via mobile now, but it is definitely not desirable in a mobile format as is.
7. Integrate with Degree Sheep and/or SIS.